



C Piscine

Rush 00

Resumen: Este documento corresponde al enunciado del Rush 00 de la C Piscine de 42.

Versión: 5.2

Índice general

I.	Instrucciones	2
II.	Introducción	4
III.	Enunciado	5
IV.	Rush 00	7
V.	Rush 01	9
VI.	Rush 02	10
VII.	Rush 03	11
VIII.	Rush 04	12
IX.	Entrega y evaluación	13

Capítulo I

Instrucciones

- El grupo será registrado AUTOMÁTICAMENTE a la evaluación
- No canceles la evaluación, no tendrás otra.
- Cualquier petición de aclaraciones adicionales sobre las instrucciones podría desembocar en una complicación posterior del enunciado.
- Debéis respetar el procedimiento de entrega para todos vuestros ejercicios.
- Este enunciado puede cambiar hasta una hora antes de la entrega.
- Los ejercicios han sido ordenados con mucha precisión del más sencillo al más complejo. En ningún caso prestaremos atención ni tendremos en cuenta un ejercicio complejo si no se ha conseguido realizar perfectamente un ejercicio más sencillo.
- El programa debe ser compilado con los flags -Wall -Wextra -Werror y utilizar gcc.
- Si vuestro programa no compila, tendréis 0.
- El programa debe estar escrito de acuerdo a la Norma. Si tiene archivos/funciones bonus, estos están incluidos en la comprobación de la Norma y la nota será 0 si la incumplen.
- Los ejercicios del rush se deben realizar en grupos de 2, 3 ó 4.
- El número del rush impuesto a vuestro grupo seguirá la siguiente regla: rango alfabético de la primera letra del login del team leader (de 1 a 26) módulo 5.
- Por lo tanto, debes realizar el proyecto junto con los miembros de tu equipo y debéis presentaros todos a la evaluación a la hora acordada.
- El proyecto debe estar terminado cuando se presente a la evaluación. El propósito de la evaluación es que presentéis y expliquéis vuestro trabajo en detalle.
- Cada miembro del grupo tendrá que estar perfectamente al corriente del trabajo realizado. Si elegís dividir el trabajo, aseguraos de que todos entendéis lo que ha hecho el resto. Durante la evaluación se harán preguntas y la nota del grupo se basará en la explicación peor.

- Evidentemente, es tu responsabilidad reunir al equipo de trabajo. Tienes todos los medios disponibles para contactar con el resto de miembros del equipo: teléfono, e-mail, paloma mensajera, sesión de espiritismo, etc. No se aceptará ninguna excusa en lo que respecta a los problemas de grupo. La vida es injusta, pero es lo que hay.
- De todas manera, si después de haberlo intentado realmente todo no puedes contactar con un miembro de tu grupo: haced el rush y entregadlo. Intentaremos encontrar una solución durante la evaluación. Incluso si el integrante que falta es el team leader, todos tenéis acceso al repositorio.
- De manera opcional, podéis contestar a varios enunciados para obtener posibles puntos extras o utilizar argumentos de programa para probar vuestra función.



Es absolutamente obligatorio haber contestado perfectamente a los enunciados obligatorios para acceder a los enunciados extras. Si un enunciado extra funciona, pero el obligatorio no, tendréis un 0.

Capítulo II

Introducción

Aquí la letra de los créditos de Pinky y Cerebro:

Pinky: Cerebro, ¿qué vamos a hacer esta noche?

Cerebro: Lo mismo que hacemos todas las noches, Pinky: ¡tratar de conquistar el mundo!

Son Pinky y Cerebro

Pinky y Cerebro

Uno es un genio

El otro no está cuerdo

De laboratorio son

Con genes insertados

Son Pinky

¡Son Pinky y Cerebro, bro, bro, bro,

bro, bro, bro, bro!

Antes del amanecer

Desarrollarán su plan

Y cuando salga el sol

El mundo conquistarán

Son Pinky y Cerebro

Pinky y Cerebro

Su motivación

Es fácil de explicar

Para probar su valor

El mundo conquistarán

Son Pinky


¡Son Pinky y Cerebro, bro, bro, bro,

bro, bro, bro, bro!

¡En lugar de conquistar el mundo, es preferible que te dediques a conquistar este rush!

Capítulo III

Enunciado

	Ejercicio: 00
	rush0X
	Directorio de entrega: <i>ex00/</i>
	Archivos a entregar: <code>main.c</code> , <code>ft_putchar.c</code> , <code>rush0X.c</code>
	Funciones autorizadas: <code>write</code>

- Archivos a entregar: `main.c`, `ft_putchar.c` y el `rush0X.c`, donde 0X corresponderá al número del rush. Por ejemplo, `rush00.c`.
- Los tres archivos serán compilados juntos.
- El archivo `ft_putchar.c` debe contener la función `ft_putchar`.
- Ejemplo de `main.c`:

```
int    main()
{
    rush(5, 5);
    return (0);
}
```

- Por lo tanto, deberéis escribir la función `rush` que reciba como parámetro dos variables de tipo entero que se llamen respectivamente `x` e `y`.
- Vuestra función `rush` tendrá que mostrar en pantalla un rectángulo de `x` caracteres de ancho e `y` caracteres de alto.
- Vuestra función no debe producir un crash ni entrar en bucle infinito.
- Vuestro `main` será modificado durante la evaluación para poder cambiar los parámetros de llamada a la función `rush`. Por ejemplo, se probarán este tipo de cosas:

```
int    main()
{
    rush(123, 42);
    return (0);
}
```

Capítulo IV

Rush 00

- `rush(5,3)` debería mostrar esto:

```
$> ./a.out
o---o
|   |
o---o
$>
```

- `rush(5, 1)` debería mostrar esto:

```
$> ./a.out
o---o
$>
```

- `rush(1, 1)` debería mostrar esto:

```
$> ./a.out
o
$>
```

- `rush(1, 5)` debería mostrar esto:

```
$> ./a.out
o
|
|
|
o
$>
```


- `rush(4, 4)` debería mostrar esto:

```
$> ./a.out
o--o
| |
| |
o--o
$>
```

Capítulo V

Rush 01

- `rush(5,3)` debería mostrar esto:

```
$>./a.out
/***\
*   *
\***/
$>
```

- `rush(5, 1)` debería mostrar esto:

```
$>./a.out
/***\
$>
```

- `rush(1, 1)` debería mostrar esto:

```
$>./a.out
/
$>
```

- `rush(1, 5)` debería mostrar esto:

```
$>./a.out
/
*
*
*
\
$>
```

- `rush(4, 4)` debería mostrar esto:

```
$>./a.out
/**\
*  *
*  *
\**/
$>
```

Capítulo VI

Rush 02

- `rush(5,3)` debería mostrar esto:

```
$> ./a.out
ABBBB
B  B
CBBBC
$>
```

- `rush(5, 1)` debería mostrar esto:

```
$> ./a.out
ABBBB
$>
```

- `rush(1, 1)` debería mostrar esto:

```
$> ./a.out
A
$>
```

- `rush(1, 5)` debería mostrar esto:

```
$> ./a.out
A
B
B
B
C
$>
```

- `rush(4, 4)` debería mostrar esto:

```
$> ./a.out
ABBA
B  B
B  B
CBBB
$>
```

Capítulo VII

Rush 03

- `rush(5,3)` debería mostrar esto:

```
$> ./a.out
ABBBC
B  B
ABBBC
$>
```

- `rush(5, 1)` debería mostrar esto:

```
$> ./a.out
ABBBC
$>
```

- `rush(1, 1)` debería mostrar esto:

```
$> ./a.out
A
$>
```

- `rush(1, 5)` debería mostrar esto:

```
$> ./a.out
A
B
B
B
A
A
$>
```

- `rush(4, 4)` debería mostrar esto:

```
$> ./a.out
ABBC
B  B
B  B
ABBC
$>
```

Capítulo VIII

Rush 04

- `rush(5,3)` debería mostrar esto:

```
$> ./a.out
ABBEC
B  B
CBBBA
$>
```

- `rush(5, 1)` debería mostrar esto:

```
$> ./a.out
ABBEC
$>
```

- `rush(1, 1)` debería mostrar esto:

```
$> ./a.out
A
$>
```

- `rush(1, 5)` debería mostrar esto:

```
$> ./a.out
A
B
B
B
C
$>
```

- `rush(4, 4)` debería mostrar esto:

```
$> ./a.out
ABBEC
B  B
B  B
CBBBA
$>
```

Capítulo IX

Entrega y evaluación

Entrega tu proyecto en tu repositorio `Git` como de costumbre. Solo el trabajo entregado en el repositorio será evaluado durante la defensa. No dudes en comprobar varias veces los nombres de los archivos para verificar que sean correctos.

Como este proyecto no es comprobado por un programa, puedes organizar tus archivos como consideres oportuno, siempre y cuando entregues los archivos obligatorios y estos cumplan con los requisitos.



Sólo necesitas entregar los archivos requeridos por el enunciado de este proyecto.